Learning Non-Metric Visual Similarity for Image Retrieval

Noa Garcia
Aston University, UK
garciadn@aston.ac.uk

George Vogiatzis
Aston University, UK
g.vogiatzis@aston.ac.uk

Abstract

Can a neural network learn the concept of visual similarity? In this work, this question is addressed by training a deep learning model for the specific task of measuring the similarity between a pair of pictures in content-based image retrieval datasets. Traditionally, content-based image retrieval systems rely on two fundamental tasks: 1) computing meaningful image representations from pixels and 2) measuring accurate visual similarity between those representations. Whereas in the last few years several methods have been proposed to find high quality image representations including SIFT [21], VLAD [15] or RMAC [40], most techniques still depend on standard metrics such as Euclidean distance or cosine similarity for the visual similarity task. However, standard metrics are independent from data and might be missing the nonlinear inner structure of visual representations. In this paper, we propose to learn a non-metric visual similarity function directly from image representations to measure how alike two images are. Experiments on standard image retrieval datasets show that results are boosted when using the proposed method over standard metrics.

1. Introduction

For humans, deciding whether two images are visually similar or not is, to some extent, a natural task. However, in computer vision, this is a challenging problem and algorithms do not always succeed in matching pictures that contain similar-looking elements. This is mainly because of the well-known *semantic gap* problem, which refers to the difference or gap between low-level image pixels and high-level semantic concepts. Estimating visual similarity is a fundamental task that seeks to break this semantic gap by accurately evaluating how alike two or more pictures are. Visual similarity is crucial for many computer vision areas including image retrieval, image classification and object recognition, among others.

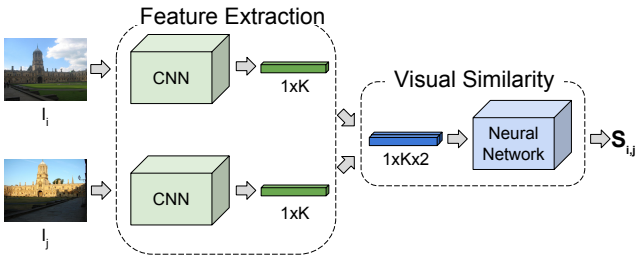


Figure 1: System overview. For any pair of images I_i and I_j, K-dimensional visual representations are extracted from a first CNN. Then, visual representation vectors are concatenated and fed into the visual similarity neural network, which outputs a score $s_{i,j}$ as the similarity estimation between the original images.

Given a query image, content-based image retrieval (CBIR) systems rank pictures in a dataset according to how similar they are with respect to the input. This can be broken into two fundamental tasks: 1) computing meaningful image representations that capture the most salient visual information from pixels and 2) measuring accurate visual similarity between these image representations to rank images according to a similarity score.

In the last years, several methods to represent visual information from raw pixels in images have been proposed, first by designing handcrafted features such as SIFT [21], SURF [3] or BRIEF [5], then by compacting these local features into a single global image descriptor using different techniques such as BOW [36], Fisher Vectors [26] or VLAD [15] and more recently by extracting deep image representations from neural networks, including Neural Codes [2], CroW [16] or RMAC [40]. However, once two images are described by feature vectors, visual similarity is commonly measured by computing a standard metric between them. Although regular distance metrics, such as Euclidean distance or cosine similarity, are fast and easy to implement, they do not take into account the possible interdependency

within the dataset, which means that even if a strong nonlinear data dependency is occurring in the visual collection, they might not be able to capture it. This suggests that learning a similarity estimation directly from visual data can improve the performance on image retrieval tasks, provided that the likely nonlinearity dependencies within the dataset are precisely learned by the similarity function.

Visual similarity learning is closely related to distance metric learning, a field of machine learning that seeks to learn a metric from data for the specific task of interest. Traditionally, distance metric learning algorithms were based on linear metrics such as the Mahalanobis distance. However, if the visual data presents any nonlinear interdependency, better results are expected when using nonlinear approaches. According to some studies [38], standard metric axioms are not valid for human perception of visual similarity and hence, visual similarity functions should not necessarily satisfy distance metric conditions. Deep learning-based similarity learning methods are mostly focused on learning an optimal mapping from pixels to a linear space in which Euclidean distance can be applied. Instead, we propose a simple approach based on neural networks to learn a non-metric similarity score in the manifold of samples in the feature space.

Figure 1 shows an overview of the proposed methodology. By training a deep learning model, we can estimate a visual similarity function that outperforms methods based on standard metric computations. One convolutional neural network (CNN) extracts image representations from a pair of input images, while a second neural network computes the visual similarity score. The visual similarity neural network is trained using both pairs of similar and dissimilar images in three stages. The output score of the similarity network can be directly applied as a similarity estimation to rank images in a CBIR system. Experimental results on image retrieval standard datasets show that our network is able to discriminate when a pair of images is similar or dissimilar and improve cosine similarity score on top of that.

The paper is laid out as follows: Section 2 introduces the related work. In Section 3, the deep architecture that learns the similarity estimation from a pair of images is described. Datasets and experimental details are presented in Section 4. Results are discussed in Section 5. Section 6 provides concluding remarks.

2. Related Work

Content-Based Image Retrieval. Content-based image retrieval searches for images by considering their visual content. Given a query image, pictures in a collection are usually ranked according to their visual similarity with respect to the query. One common approach is to represent the local visual content of images by a set of hand-crafted features, such as SIFT [21], SURF [3] or BRIEF [5]. As a single image may contain hundreds of these hand-crafted features, aggregation methods like bag-of-words (BOW) [36], Fisher Vectors [26] or VLAD [15] are typically used to encode local features into a global compact vector, thereby improving computational efficiency and scalability. Recently, because of the latest advancements on deep learning, features obtained from deep learning methods such as convolutional neural networks (CNN) have rapidly become the new state-of-the-art in image retrieval.

Deep Learning for Image Retrieval. Instead of hand-crafted features, deep image retrieval uses activations from CNNs as image representations. At first, related work [2, 34, 41, 20] proposed to extract these image representations from one of the last fully connected layers of a network trained on ImageNet [32], a popular image classification dataset. When deeper networks such as GoogLeNet [37] and VGG [35] appeared, some authors [1, 45, 34, 44] proposed to use mid-layer representations obtained from the convolutional layers instead of the fully connected activations. Since then, there have been several attempts to aggregate these high-dimensional convolutional representations into a compact vector. For example, [10, 45] compacted deep features by using VLAD, [24] encoded the neural codes into an histogram of words, [1, 16] applied sum-pooling to obtain a compact representation and [31, 40] aggregated deep features by max-pooling them into a new vector. A different approach is to train the network to directly learn compact binary codes end-to-end [8, 19]. Finally, some authors have shown that fine-tunning the networks with similar data to the target task increases the performance significantly [2, 11, 30, 33].

These methods are focused on finding high quality features to represent visual content efficiently. Then, visual similarity is computed by simply applying a standard metric distance to these representations. General metrics, such as Euclidean distance or cosine similarity, however, might be failing to consider the inner data structure of these visual representations. Learning a similarity function directly from data may help to capture the human perception of visual similarity in a better way.

Similarity Learning. Some of the most popular similarity learning works proposed so far, such as OASIS [6] and MLR [23], are based on linear metric learning by optimizing the weights of a linear transformation matrix. Although linear methods are easier to optimize and less prone to overfitting, nonlinear algorithms are expected to achieve higher accuracy modeling the possible nonlinearities of data. Nonlinear similarity learning based on deep learning has been recently applied to many different visual contexts. In low-level image matching, CNNs have been trained to match pairs of patches for stereo matching [46, 22] and optical flow [9, 39]. In high-level image matching, deep learning techniques have been proposed to learn low-dimensional

embedding spaces in face verification [7], retrieval [43, 42], classification [13, 29, 25] and product search [4], either by using siamese [7] or triplet [42] architectures.

In general, these methods rely on learning a mapping from image pixels to a low dimensional target space to compute the final similarity decision by using a standard metric. They are designed to find the best projection in which a linear distance can be successfully applied. Instead of projecting the visual data into some linear space, that may or may not exist, our approach seeks to learn the non-metric visual similarity score itself. Similarly, [18] and [12] used a CNN to decide whether or not two input images are a match, applied to pedestrian reindentification and patch matching, respectively. In these methods, the networks are trained as a binary classification problem (i.e. same or different pedestrian/patch), whereas in an image retrieval ranking problem, a regression score is required. Inspired by the results of [41], which showed that combining deep features with similarity learning techniques can be very beneficial for the performance of image retrieval systems, we propose to train a deep learning algorithm to learn non-metric similarities for image retrieval and improve results in top of high quality image representation methods.

3. Learning Visual Similarity

Here, we describe a deep architecture that learns the mapping from a pair of images to a similarity score.

3.1. Definition

Visual similarity is the task that measures how related two images are by using their visual content. Given n samples in the training image collection I, for each image $I_i \in I$ with $i \in [1, n]$, a global d-dimensional representation $x_i \in \mathbb{R}^d$ is obtained as $x_i = f(I_i, w_f)$, where f is the function that maps images into global features and w_f is the set of parameters of f. We define $s_{i,j}$ as the similarity score which measures how alike two images I_i and I_j are. The higher $s_{i,j}$ is, the more similar I_i and I_j are. The aim is to learn a visual similarity function S that computes the similarity score from global image representations as:

$$s_{i,j} = S(x_i, x_j) = g(f(I_i, w_f), f(I_j, w_f), w_g) \quad (1)$$

where g is a nonlinear function and w_g is the set of parameters to optimize.

Note that g does not have to be a metric in order to be a similarity function and thus, it is not required to satisfy the rigid constraints of metric axioms, i.e. non-negativity, identity of indiscernibles, symmetry and triangle inequality. Some non-metric similarity works such as [38] suggest that these restrictions are not compatible with human perception. As an example, they showed that although a centaur might be visually similar to both a person and a horse, the person and the horse are not similar to each other. A possible explanation for this phenomenon is that when comparing two images, human beings may pay more attention to similarities and thus, similar portions of the images may be more discriminative than dissimilar parts. To overcome the issues associated with applying strong rigid constraints to visual similarity, we propose to learn the non-metric similarity function g using a neural network approach.

3.2. Image Representation

First, we describe the image representation method f we use in our system. As this work aims to learn a non-metric similarity estimation from visual data, our efforts are not focused on improving existing image representation methods, but to learn how to compare them. Without loss of generality, we use the state-of-the-art RMAC descriptor proposed in [40] as image representation method, although any other image representation method can be considered as well.

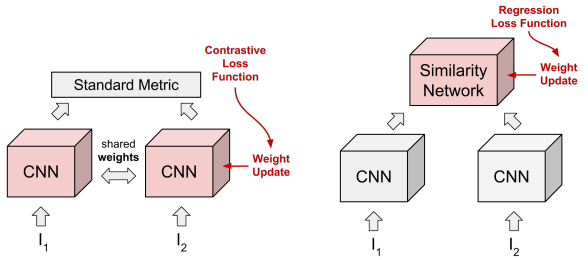
RMAC or Regional Maximum Activations of Convolution is a deep global image representation obtained from the last convolutional layer of a pre-trained CNN on ImageNet classification task [32]. When an image is fed into the network, the last convolutional layer outputs a $W \times H \times K$ response, where K is the number of filters and W and H are the spatial width and height of the output, respectively, that depend on the network architecture as well as on the size of the input image. The response of the k-th filter of the last convolutional layer can be represented by Ω_k, a 2D tensor of size $W \times H$. If $\Omega_k(p)$ is the response at a particular position p, and R is a spatial region within the feature map, the regional feature vector f_R is defined as:

$$f_R = [f_{R,1} \dots f_{R,k} \dots f_{R,K}]^\top \quad (2)$$

where $f_{R,k} = \max_{p \in R} \Omega_k(p)$. Thus, f_R consists of the maximum activation of each filter inside the region R. Several regional features are extracted at different multi-scale overlapping regions. Each of these regional vectors is independently post-processed with $\ell 2$-normalization, PCA-whitening and $\ell 2$-normalization, as suggested in [14]. Finally, regional vectors are summed and $\ell 2$-normalized once again to obtain the final compact vector. The size of the final vector is K, which is independent of the size of the input image, its aspect ratio or the number of regions used.

3.3. Similarity Network

To compare two images and obtain their visual similarity score we learn the similarity function g by training a deep learning architecture. Given two input images I_i and I_j, we first extract their image representations x_i and x_j, respectively, as explained in Section 3.2. Then, the two K-dimensional global vectors are concatenated along the third dimension (i.e. depth) and fed into the similarity network, as shown in Figure 1. This process is different to the



(a) Siamese network (b) Similarity network

Figure 2: Differences between a siamese architecture (a), where the network is trained to map pixels into high-quality vector representations, and our architecture (b), where the network is trained to learn a similarity function in top of vector representations. Moreover, in the similarity network, weights are not necessarily shared.

standard siamese architecture [7] because the latest maps images into vector representations and updates the shared weights according to the learning protocol and our approach trains and updates the similarity network on top of high-quality vector representations. Figure 2 shows the difference between both approaches.

The similarity network is composed by a set of fully connected layers, each one of them followed by a ReLU nonlinearity layer [17]. The input of the network is fixed to be of $1 \times K \times 2$ size, so the size of the first layer, as it is fully connected, is $1 \times K \times 2 \times Ch$, where Ch is the number of channels. The hidden layers are of size $1 \times Ch \times 2 \times Ch$. On the other hand, the output layer is of size $1 \times Ch \times 2 \times 1$ and it is not followed by a ReLU layer, as the output score is expected to cover a full range of values, both positive and negative. Finally, during training, we add an ℓ_1-norm loss function layer to compute the regression loss L as:

$$L(I_i, I_j) = |s_{i,j} - y_{i,j}| = |g(x_i, x_j, w_g) - y_{i,j}| \quad (3)$$

where $s_{i,j}$ is the network output and $y_{i,j}$ is the annotated score. Four configurations A-D with different number of filters Ch and number of hidden layers are proposed and tested during our experiments (Table 1).

3.4. Training Similarity

The visual similarity network is trained in three stages. In each stage the weights are initialized by the trained weights of the previous stage while the learned task gets progressively more difficult.

Stage 1: Standard Metric

In the first stage, we use the network to learn a standard similarity function based on the cosine similarity. We generate random pairs of vectors, x_i and x_j, and we assign the cosine similarity between them as the score label $y_{i,j}$:

$$y_{i,j} = \frac{x_i \cdot x_j}{\|x_i\|\|x_j\|} \quad (4)$$

In order to train the model in the full range of possible values, pairs are produced so that the cosine similarity is uniformly distributed within the training set.

Table 1: Network architectures A-D. Fully connected layers denoted as FC-{number of filters}. Last row is the number of parameters (in millions) of each configuration.

A	B	C	D
FC-1024	FC-4096	FC-8192	FC-4096
ReLU	ReLU	ReLU	ReLU
FC-1024	FC-4096	FC-8192	FC-4096
ReLU	ReLU	ReLU	ReLU
-	-	-	FC-4096
-	-	-	ReLU
FC-1	FC-1	FC-1	FC-1
2.1	21	76	38

Stage 2: Visual Similarity

In the second stage, the basic similarity network learns to increase the similarity score when given two matching images and to decrease it when a pair of images is not a match. The weights in this training stage are initialized by the weights obtained during Stage 1. We now use pairs of image representation vectors x_i and x_j, randomly chosen from our training image dataset. The score label is set to:

$$y_{i,j} = \begin{cases} \frac{x_i \cdot x_j}{\|x_i\|\|x_j\|} + \Delta, & \text{if } x_i \text{ and } x_j \text{ are similar} \\ \frac{x_i \cdot x_j}{\|x_i\|\|x_j\|} - \Delta, & \text{otherwise} \end{cases} \quad (5)$$

where Δ is the margin parameter. Thus, the model learns to discriminate when a pair of images are similar (dissimilar) and assigns it a higher (lower) value than the standard score.

In this stage, the model learns how to compute a similarity score from examples of images that are known to be matching or non-matching. Therefore a relevant dataset to the final retrieval task should be used. Similarity between pairs might be decided using different techniques, such as image classes, score based on local features or manual labeling, among others. Without loss of generality, we consider two images as similar when they belong to the same class and as dissimilar when they belong to different classes.

Figure 3: Difficult pairs misclassified after Stage 2. (Upper) Lower row: (dissimilar) similar images in which the network score is (lower) higher than the cosine similarity.

Stage 3: Hard Examples

In the final stage, the similarity network is refined by training it specifically by using *difficult pairs* of images. Previous works [11, 30] have shown that fine-tunning neural networks using difficult samples is very helpful in terms of performance. This is easy to understand: if the network is only trained by using *easy pairs* (e.g. a car and a dog), it will not be able to discriminate between *difficult pairs* (e.g. a car and a van). To choose the set of hard pairs we compute the scores of a random set of image pairs by using the network trained in Stage 2. Those pairs in which the network output is worse than the cosine similarity measure are selected as difficult pairs for retraining[1]. Examples of difficult image pairs can be seen in Figure 3.

4. Experiments

In this section, we present the datasets and the experimental details used in our experiments.

4.1. Testing Datasets

Our approach is evaluated on the standard image retrieval datasets described below.

Oxford5k [27]: a dataset that consists of 5,062 images of 11 different Oxford landmarks. The query set contains 55 annotated images, 5 per landmark.

Paris6k [28]: a datasets that consists of 6,412 images of 11 different Paris landmarks. The query set contains 55 annotated images, 5 per landmark.

Land5k: a validation subset of the Landmarks database [2]. It consists of the 4,915 validation images from 529 classes. A random selection of 45 images is used as queries.

Oxford105k, **Paris106k**: the large-scale versions of Oxford5k and Paris6k, respectively. They include 100,000 distractor images from Flickr [27].

In both the Oxford5k and the Paris6k collections query images are cropped according to the region of interest provided by the authors of the datasets. Evaluation is performed by computing the mean Average Precision (mAP), using the provided ground truth and algorithms. For Landmarks5k we consider an image to be relevant to the query when it belongs to the same class.

4.2. Training Datasets

For the purposes of this work, having a training dataset as similar as possible to the final similarity task is essential. We create several versions of the training dataset to evaluate the effect of using different samples in the training process.

Landmarks-clean [11]: an automatically cleaned subset of Landmarks [2] dataset which officially contains about 49,000 images from 586 landmarks. However, due to broken URLs, we could only download 33,119 training images and 4,915 validation images. This dataset does not contain images from classes that overlap with Oxford5k and Paris6k datasets as they were manually removed.

Landmarks-clean-extra: the Landmarks-clean collection in addition to about 500 images from Oxford5k and 1,700 images from Paris6k classes. In total, it contains 35,342 training images belonging to 605 different landmarks. Note that query images are not added in any case and they remain unseen by the system.

Landmarks-clean-extra-500: the Landmarks-clean collection plus 250 random images from each of the Oxford5k and Paris6k datasets. In total, it contains 33,619 training images.

4.3. Experimental Details

Image Representation. To compute RMAC representations we use the VGG16 network [35], which has been previously pre-trained on the ImageNet dataset [32]. Unless otherwise stated, we use the default values proposed in [40] to obtain 512-dimensional RMAC vectors. VGG16 network is used off-the-shelf without any retraining or fine-tunning performed on top of it. Experimental results have shown that RMAC representations are very sensitive to the PCA matrices used in the post-processing step. As we are keeping query images unseen by the system and not using them in the PCA matrices computation as in [40], our results are slightly different to theirs.

Visual Similarity Learning. Similarity learning is trained using almost a million of random pairs, of which half of the pairs are visual matches and the other half are non-matches. PCA whitening is done using Paris5k images.

[1] A worse score is a score that is lower in the case of a match and higher in the case of a non-match.

As RMAC representation performs better in high resolution images, we re-scale all the images up to 1024 pixels, keeping the original aspect ratio of the pictures. For the similarity network, four different configurations A-D (Table 1) are explored during our experiments. The network is optimized using backpropagation and stochastic gradient descent. We use a learning rate of 0.001, a batch size of 100, a weight decay of 0.0005 and momentum of 0.9.

Computational cost. Standard metrics are relatively fast and computationally cheap. Our visual similarity network involves the use of millions of parameters that inevitable increase the computational cost. However, it is still feasible to compute in a reasonable amount of time. In our experiments, training time is about 5 hours in a GeForce GTX 1080 GPU and testing time for a pair of images is 1.25 ms on average (0.35 ms when using cosine similarity).

5. Results

In this section, we present the experimental results when visual similarity is learned with our method.

5.1. Architecture Discussion

As shown in Table 1 four different configurations A-D for the similarity neural network are proposed. We compare the performance of each one during Stage 1, when the network is trained with the standard cosine similarity measurement. If s_l is the network score and y_l is the cosine similarity of the l-th pair with $l = 1..L$, we evaluate each network by computing the mean squared error, MSE, and the correlation coefficient, ρ, as:

$$MSE = \frac{1}{L}\sum_{l=1}^{L}(s_l - y_l)^2 \tag{6}$$

$$\rho = \frac{1}{L-1}\sum_{l=1}^{L}\frac{s_l - \mu_s}{\sigma_s}\frac{y_l - \mu_y}{\sigma_y} \tag{7}$$

where μ_s and σ_s are the mean and standard deviation of the vector of network scores s, respectively, and μ_y and σ_y are the mean and standard deviation of the vectors of cosine similarities y, respectively.

Results can be seen in Table 2. Unsurprisingly, the configuration with bigger number of parameters, C, achieves the best MSE and ρ results, both in training and validation sets. However, the performance of networks B and D is very close to the performance of network C. As network B requires only 21 million parameters and network C requires 76 million parameters, we keep configuration B as our default architecture for the rest of the experiments.

5.2. Evaluation of the similarity network

To evaluate our similarity network, we compute the mAP at each stage of the training process (Section 4.2). Results when using different training datasets can be found in Table 3. Cosine similarity is computed as a baseline. We denote as DeepCosine the results obtained after the first stage, when the network is trained to mimic cosine similarity. Naturally, DeepCosine performs worse than the cosine similarity, as it is an estimation of the cosine metric. DeepSim refers to the results obtained after the second stage, when the network is fine-tunned to learn visual similarity with random pairs of images. DeepSimH are the results after the last stage, when the network is trained by using both random and hard pairs of images. We compare our approach against the standard similarity learning algorithm OASIS [6].

Table 2: MSE and ρ in networks A-D. Training: 22.5 million random pairs. Validation: 7.5 million random pairs.

	Training Data		Validation Data	
	MSE	ρ	MSE	ρ
A	0.00021	0.946	0.00035	0.909
B	0.00008	0.978	0.00019	0.965
C	**0.00007**	**0.982**	**0.00012**	**0.974**
D	0.00009	0.978	0.00019	0.964

Out similarity networks outperform OASIS in all the testing datasets. Moreover when using Landmarks-clean-extra as training dataset, results are boosted with respect to the standard metric, achieving improvements ranging from 20% (Oxford5k) to 40% (Pairs6k). When using a small subset of images from Oxford5k and Paris6k classes, i.e. Landmarks-clean-extra-500 dataset, our similarity networks also improve mAP with respect to the cosine similarity in the three testing datasets. This indicates that visual similarity can be learnt even when using a reduced subset of the target image domain. However, visual similarity does not transfer well across domains when no images of the target domain are used during training. For example, when training with Landmarks-clean dataset, the visual similarity estimated using either OASIS and our DeepSim and DeepSimH only improves over cosine similarity in the Land5k testing scenario, as both training and testing images belongs to the same domain of image classes.

These results show that our network is able to learn whether two images are similar or not and improve the cosine similarity metric accordingly. Figure 4 shows how the mAP is affected when using DeepSimH network and choosing different values of Δ. Except when $\Delta = 0$ (i.e. visual similarity is not learned), mAP is always improved with respect to the standard cosine similarity metric.

Table 3: mAP when using different training datasets and several Δ values.

	Oxford5k	Paris6k	Land5k
Cosine Similarity	0.665	0.638	0.564
DeepCosine	0.638	0.596	0.549
	Landmarks-clean		
OASIS [6]	0.514	0.385	0.578
DeepSim (Δ 0.2)	**0.658**	0.460	0.669
DeepSimH (Δ 0.2)	0.655	0.503	0.697
DeepSimH (Δ 0.4)	0.637	0.504	0.737
DeepSimH (Δ 0.6)	0.613	**0.514**	0.776
DeepSimH (Δ 0.8)	0.600	0.511	**0.783**
	Landmarks-clean-extra-500		
OASIS [6]	0.570	0.651	0.589
DeepSim (Δ 0.2)	0.717	0.654	0.671
DeepSimH (Δ 0.2)	**0.719**	0.677	0.693
DeepSimH (Δ 0.4)	0.703	0.701	0.745
DeepSimH (Δ 0.6)	0.703	**0.716**	0.776
DeepSimH (Δ 0.8)	0.685	0.710	**0.803**
	Landmarks-clean-extra		
OASIS [6]	0.619	0.853	0.579
DeepSim (Δ 0.2)	0.718	0.757	0.668
DeepSimH (Δ 0.2)	0.786	0.860	0.662
DeepSimH (Δ 0.4)	0.794	0.878	0.706
DeepSimH (Δ 0.6)	0.789	0.885	0.735
DeepSimH (Δ 0.8)	**0.808**	**0.891**	**0.758**

5.3. Comparison with the state of the art

Finally, we compare our method against several state-of-the-art techniques (Table 4). As standard practice, works are split into two main groups: off-the-shelf and fine-tunning approaches. Off-the-shelf are techniques that extract visual representations by using CNNs trained on ImageNet dataset [32] without modifying the network. On the other hand, fine-tunning methods retrain the network to compute more accurate visual representation. For a fair comparison, we do not consider methods that apply query expansion or image re-ranking. When using off-the-shelf RMAC features,

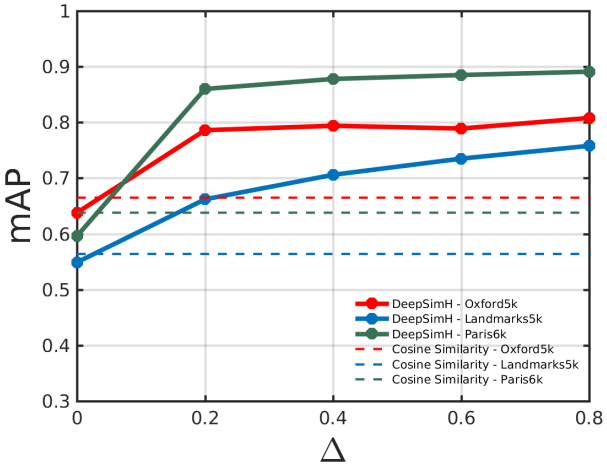


Figure 4: mAP versus Δ. Rigid lines are DeepSimH scores, dashed lines are cosine similarity scores.

our DeepSimH approach outperforms previous methods in every dataset. To compare against fine-tunned methods, we compute RMAC vectors using the fine-tunned version of VGG16 proposed in [30] and training our DeepSimH exactly in the same way as in the off-the-shelf version. Accuracy is significantly improved when using our similarity network instead of the analogous cosine similarity method [30]. DeepSimH achieves the best mAP precision in three out of four datasets, and comes second in the fourth.

6. Conclusions

We have presented a method for learning visual similarity directly from visual data. Instead of using a rigid metric distance, such as the standard cosine similarity, we propose to train a neural network model to learn a similarity estimation between a pair of visual representations previously extracted from input images. Our method outperforms state-of-the-art approaches based on rigid distances in standard image retrieval collection of images and experimental results showed that learning a non-metric visual similarity function is beneficial in image retrieval tasks.

We end with some interesting open questions. Firstly, as the entirety of our similarity computation is now based on a network architecture, this opens the possibility of true end-to-end training for content based image retrieval. Another question concerns efficient computation of exact or approximate K-nearest neighbours based on the learned network similarity functions. These questions are the subject of planned future work.

Table 4: mAP results for different state-of-the-art techniques. Dim is the dimensionality of the image representation. Similarity is the similarity method used to rank dataset images. * symbol is used to denote our implementation. In this table, we use $\Delta = 0.8$. Approaches using off-the-shelf and fine-tunned features are showed separately.

	Method	Dim	Similarity	Oxford5k	Oxford105k	Paris6k	Paris106k
Off-the-shelf	Neural Codes [2]	512	L2	0.435	0.392	-	-
	Razavian *et al*. [34]	4096	Averaged L2	0.322	-	0.495	-
	Wan *et al*. [41]	4096	OASIS [6]	0.466	-	0.867	-
	SPoC [1]	256	Cosine	0.657	0.642	-	-
	DeepIndex [20]	28k	Inverted Index	-	-	0.812	-
	Ng *et al*. [45]	128	L2	0.593	-	0.59	-
	CroW [16]	512	L2	0.708	0.653	0.797	0.722
	Mohedano *et al*. [24]	25k	Cosine	0.739	0.593	0.82	0.648
	Salvador *et al*. [33]	512	Cosine	0.588	-	0.656	-
	RMAC [40]	512	Cosine	0.669	0.616	0.83	0.757
	RMAC+DeepSimH *	512	DeepSimH	**0.808**	**0.772**	**0.891**	**0.818**
Fine-tunning	Neural Codes [2]	512	L2	0.557	0.522	-	-
	Gordo *et al*. [11]	512	Cosine	0.831	0.786	0.871	0.797
	Wan *et al*. [41]	4096	OASIS [6]	0.783	-	**0.947**	-
	Radenovic *et al*. [30]	512	Cosine	0.77	0.692	0.838	0.764
	Salvador *et al*. [33]	512	Cosine	0.71	-	0.798	-
	[30]+DeepSimH *	512	DeepSimH	**0.882**	**0.821**	0.882	**0.829**

References

[1] A. Babenko and V. Lempitsky. Aggregating local deep features for image retrieval. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1269–1277, 2015. 2, 8

[2] A. Babenko, A. Slesarev, A. Chigorin, and V. Lempitsky. Neural codes for image retrieval. In *Proceedings of the IEEE European Conference on Computer Vision*, pages 584–599, 2014. 1, 2, 5, 8

[3] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *European Conference on Computer Vision*, pages 404–417. Springer, 2006. 1, 2

[4] S. Bell and K. Bala. Learning visual similarity for product design with convolutional neural networks. *ACM Transactions on Graphics (TOG)*, 34(4):98, 2015. 3

[5] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. Brief: Binary robust independent elementary features. In *European Conference on Computer Vision*, pages 778–792. Springer, 2010. 1, 2

[6] G. Chechik, V. Sharma, U. Shalit, and S. Bengio. Large scale online learning of image similarity through ranking. *Journal of Machine Learning Research*, 11(Mar):1109–1135, 2010. 2, 6, 7, 8

[7] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Proceedings of the IEEE Conference on Computer Vision*

and Pattern Recognition, pages 539–546, 2005. 3, 4

[8] V. Erin Liong, J. Lu, G. Wang, P. Moulin, and J. Zhou. Deep hashing for compact binary codes learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2475–2483, 2015. 2

[9] P. Fischer, A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, 2015. 2

[10] Y. Gong, L. Wang, R. Guo, and S. Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. In *Proceedings of the IEEE European Conference on Computer Vision*, pages 392–407, 2014. 2

[11] A. Gordo, J. Almazán, J. Revaud, and D. Larlus. Deep image retrieval: Learning global representations for image search. In *Proceedings of the IEEE European Conference on Computer Vision*, pages 241–257, 2016. 2, 5, 8

[12] X. Han, T. Leung, Y. Jia, R. Sukthankar, and A. C. Berg. Matchnet: Unifying feature and metric learning for patch-based matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3279–3286, 2015. 3

[13] E. Hoffer and N. Ailon. Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition*, pages 84–92, 2015. 3

[14] H. Jégou and O. Chum. Negative evidences and cooccurences in image retrieval: The benefit of pca and whitening. In *Computer Vision–ECCV 2012*, pages 774–787. Springer, 2012. 3

[15] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3304–3311, 2010. 1, 2

[16] Y. Kalantidis, C. Mellina, and S. Osindero. Cross-dimensional weighting for aggregated deep convolutional features. In *Proceedings of the IEEE European Conference on Computer Vision*, pages 685–701, 2016. 1, 2, 8

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 4

[18] W. Li, R. Zhao, T. Xiao, and X. Wang. Deepreid: Deep filter pairing neural network for person re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 152–159, 2014. 3

[19] K. Lin, H.-F. Yang, J.-H. Hsiao, and C.-S. Chen. Deep learning of binary hash codes for fast image retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 27–35, 2015. 2

[20] Y. Liu, Y. Guo, S. Wu, and M. S. Lew. Deepindex for accurate and efficient image retrieval. In *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval*, pages 43–50. ACM, 2015. 2, 8

[21] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004. 1, 2

[22] W. Luo, A. G. Schwing, and R. Urtasun. Efficient deep learning for stereo matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5695–5703, 2016. 2

[23] B. McFee and G. R. Lanckriet. Metric learning to rank. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 775–782, 2010. 2

[24] E. Mohedano, K. McGuinness, N. E. O'Connor, A. Salvador, F. Marqués, and X. Giró-i Nieto. Bags of local convolutional features for scalable instance search. In *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval*, pages 327–331. ACM, 2016. 2, 8

[25] H. Oh Song, Y. Xiang, S. Jegelka, and S. Savarese. Deep metric learning via lifted structured feature embedding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4004–4012, 2016. 3

[26] F. Perronnin, Y. Liu, J. Sánchez, and H. Poirier. Large-scale image retrieval with compressed fisher vectors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3384–3391, 2010. 1, 2

[27] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007. 5

[28] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008. 5

[29] Q. Qian, R. Jin, S. Zhu, and Y. Lin. Fine-grained visual categorization via multi-stage metric learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3716–3724, 2015. 3

[30] F. Radenović, G. Tolias, and O. Chum. Cnn image retrieval learns from bow: Unsupervised fine-tuning with hard examples. In *Proceedings of the IEEE European Conference on Computer Vision*, pages 3–20, 2016. 2, 5, 7, 8

[31] A. S. Razavian, J. Sullivan, S. Carlsson, and A. Maki. Visual instance retrieval with deep convolutional networks. *ITE Transactions on Media Technology and Applications*, 4(3):251–258, 2016. 2

[32] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 2, 3, 5, 7

[33] A. Salvador, X. Giró-i Nieto, F. Marqués, and S. Satoh. Faster r-cnn features for instance search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 9–16, 2016. 2, 8

[34] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 806–813, 2014. 2, 8

[35] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, 2014. 2, 5

[36] J. Sivic, A. Zisserman, et al. Video google: A text retrieval approach to object matching in videos. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1470–1477, 2003. 1, 2

[37] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015. 2

[38] X. Tan, S. Chen, J. Li, and Z.-H. Zhou. Learning non-metric partial similarity based on maximal margin criterion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 168–145, 2006. 2, 3

[39] J. Thewlis, S. Zheng, P. Torr, and A. Vedaldi. Fully-trainable deep matching. In *Proceedings of the British Machine Vision Conference*, 2016. 2

[40] G. Tolias, R. Sicre, and H. Jégou. Particular object retrieval with integral max-pooling of cnn activations. In *Proceedings of the International Conference on Learning Representations*, 2016. 1, 2, 3, 5, 8

[41] J. Wan, D. Wang, S. C. H. Hoi, P. Wu, J. Zhu, Y. Zhang, and J. Li. Deep learning for content-based image retrieval: A comprehensive study. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 157–166. ACM, 2014. 2, 3, 8

[42] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu. Learning fine-grained image similarity with deep ranking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1386–1393, 2014. 3

[43] P. Wu, S. C. Hoi, H. Xia, P. Zhao, D. Wang, and C. Miao. Online multimodal deep similarity learning with application to image retrieval. In *Proceedings of the 21st ACM international conference on Multimedia*, pages 153–162. ACM, 2013. 3

[44] L. Xie, R. Hong, B. Zhang, and Q. Tian. Image classification and retrieval are one. In *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval*, pages 3–10. ACM, 2015. 2

[45] J. Yue-Hei Ng, F. Yang, and L. S. Davis. Exploiting local features from deep networks for image retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 53–61, 2015. 2, 8

[46] S. Zagoruyko and N. Komodakis. Learning to compare image patches via convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4353–4361, 2015. 2